Architecture Overview

Here is a brief presentation given in Feb-2008 at NASA Ames : EUROPA ASR Talk

The modules of EUROPA and their interdependencies are set out in Figure 1. EUROPA is a highly modular architecture and modules can be developed, tested and applied quite independently.

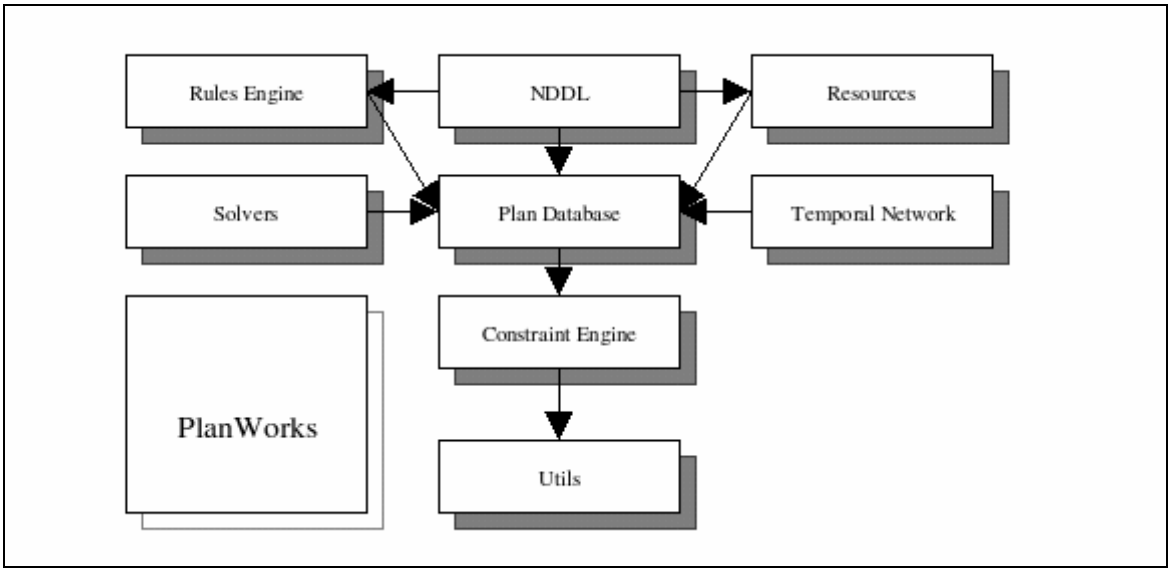

Figure 1 EUROPA Modules and their dependencies

1. The *Utils* module provides common C++ utility classes for error checking, smart pointers etc. It also includes a very useful debugging utility. Many common programming practices in EUROPA development are built on assets in this module.
2. The *Constraint Engine* is the nexus for consistency management. It provides a general-purpose component-based architecture for handling dynamic constraint networks. It deals in variables and constraints. It includes an open propagation architecture making it straightforward to integrate specialized forms of local and global constraint propagation.
3. The *Plan Database* adds higher levels of abstractions for tokens and objects and the interactions between them. This is the code embodiment of the EUROPA planning paradigm. It supports all services for creation, deletion, modification and inspection of partial plans. It maintains the dynamic constraint network underlying a partial?plan by delegation to the Constraint Engine and leverages that propagation infrastructure to maintain relationships between tokens and objects.
4. The *Solvers* module provides abstractions to support search in line with the EUROPA planning approach. It includes a component-based architecture for *Flaw Identification*, *Resolution* and *heuristics* as well as an algorithm for chronological backtracking search. As additional search algorithms are implemented they will be added to this module.
5. The *Rules Engine* module provides the inference capabilities based on domain rules described in the model. It is almost exclusively used to execute NDDL rules but can be extended for custom rule formats.
6. The *Resources* module provides specialized algorithms and data structures to support metric resources (e.g. battery, power bus, disk drive).
7. The *Temporal Network* module provides specialized algorithms and data structures to support efficient propagation of temporal constraints.
8. The *NDDL* module provides a parser and compiler for NDDL (pronounced noodle) which is a very high-level, object-oriented, declarative domain and problem description language. This module defines the

mapping from the language to the code and consequently interfaces to a number of key modules in the system.

9. *PlanWorks* is a java application for visualization and debugging of plans and planning. It is loosely coupled to the other EUROPA modules through a JNI interface.

From an application developer?s view-point, the modules of interest are: *NDDL*, *Solvers* and *PlanWorks*. These modules address modeling, search and troubleshooting respectively. Other modules will be explored in the context of making customized extensions.